An Adaptive VNS and Skewed GVNS Approaches for School Timetabling Problems

Ulisses Rezende Teixeira[1], Marcone Jamilson Freitas Souza[2], Sérgio Ricardo de Souza[1(✉)], and Vitor Nazário Coelho[3]

[1] Federal Center of Technological Education of Minas Gerais (CEFET-MG), Av. Amazonas, 7675, Belo Horizonte, MG 30510-000, Brazil
ulisses.rezende@gmail.com, sergio@dppg.cefetmg.br
[2] Federal University of Ouro Preto (UFOP), Campus Universitário, Morro do Cruzeiro, Ouro Preto, MG 35400-000, Brazil
marcone@ufop.edu.br
[3] Fluminense Federal University (UFF), Rua Miguel de Frias, 9, Niterói, RJ 24220-900, Brazil
vncoelho@gmail.com

Abstract. The School Timetabling Problem is widely known and it appears at the beginning of the school term of the institutions. Due to its complexity, it is usually solved by heuristic methods. In this work, we developed two algorithms based on the Variable Neighborhood Search (VNS) metaheuristic. The first one, named Skewed General Variable Neighborhood Search (SGVNS), uses Variable Neighborhood Descent (VND) as local search method. The second one, so-called Adaptive VNS, is based on VNS and probabilistically chooses the neighborhoods to do local searches, with the probability being higher for the more successful neighborhoods. The computational experiments show a good adherence of these algorithms for solving the problem, especially comparing them with previous works using the same metaheuristic, as well as with previous published results of the winning algorithm of the International Timetabling Competition of 2011.

Keywords: School Timetabling · Variable Neighborhood Search · SGVNS · International Timetabling Competition

1 Introduction

The task of creating a school timetabling consists, in a very simplified way, in determining for each class and time slot the school subject and its respective teacher. This is a very hard activity and it may take a lot of hours or even days depending on the amount of classes, time slots, and teachers [2]. This combination of class, teacher, and subject follows some rules or constraints. The compliance according to these constraints determines if a solution is feasible or not and how good it is.

A. Sifaleras et al. (Eds.): ICVNS 2018, LNCS 11328, pp. 101–113, 2019.
https://doi.org/10.1007/978-3-030-15843-9_9

The constraints are usually divided into two groups [20]:

(i) Hard constraints: they are mandatory constraints. If they are not met, the solution is infeasible. For example, if one teacher is allocated to more than one class at the same time-slot, the solution is invalid;
(ii) Soft constraints: these are non-mandatory restrictions. They should be met only when possible but when this is not the case, the solution still remains feasible. For example, no occurrence of idle time in a timetabling of a specific teacher is expected, but the existence of it does not infeasible the solution.

Since the pioneer work of Gotlieb [8], many techniques have been used to solve timetabling problems. According to [21] and [19], this interest is due to three main points:

(i) Difficulty to find a solution: in view of the big amount of constraints, the goal of finding a feasible solution is a hard task and it may takes many days of manual work due to the amount of involved resources (classes, teachers, time slots);
(ii) Practical importance: to build a timetabling is a basic necessity of all educational institutions. A good school timetabling can impact the life of a big quantity of people, especially students and teachers. It can impact the efficiency of the classes and student's performance too;
(iii) Theoretical importance: school timetabling is a NP-hard problem [7]. Thus, it is challenging to develop efficient algorithms to solve it.

The interest of the academic community in seeking more efficient solution methods for solving the problem grew especially in the late 1990s and early 2000s. As a result, an international conference, called PATAT (Practice and Theory on Automated Timetabling), was created. This conference originated specific competitions called ITC (*International Timetabling Competition*), the most recent of them was organized in 2011.

Besides that, specialists in the School Timetabling's class of problem created a standard to represent it, called XHSTT, as well as a library specialized to manipulate their instances, called KHE.

In this paper, the school timetabling problem is approached using the VNS metaheuristic in two variances: skewed VNS and adaptive VNS. In both approaches, it is used the KHE library and the instances from ITC 2011. The results were compared with the algorithm Goal Solver, winner of ITC 2011, the last competition specialized in problems from this kind.

The paper is organized as follows: Sect. 2 presents the KHE library, the XHSTT format and the ITC 2011; Sect. 3 presents the proposed algorithms and Sect. 4 shows their results. Finally, on Sect. 5 the conclusions and some future works are presented.

2 Context

The researches in school timetabling had an impulse with the organization of specialized competitions of algorithms to solve this type of problem (such as the

ITC, described at Sect. 2.3), as well as the creation of a standard to represent and treat instances (the XHSTT, described at Sect. 2.1) and the creation of a library to handle this format (the KHE, described at Sect. 2.2).

2.1 XHSTT Standard

The XHSTT format [17], an acronym to XML for High School TimeTabling, is a format based on XML's markup language that establishes specific structures to treat resources, time-slots and their respective constraints.

This format is divided in three basic entities:

(i) Time and resource: the time entity consists of a time-slot or a set of time-slots and the resources are subdivided into three subcategories: students, teachers, and rooms;
(ii) Events: an event is the basic unit of assignment, representing a simple lesson;
(iii) Constraints: it is responsible to determine the distribution of resources in the events. It can be defined by hard or soft constraints, according to the criteria expected for a specific solution to be feasible or infeasible. Besides, it is subdivided into three subcategories: (i) basic constraints of schedule; (ii) constraints of events; and (iii) constraints of resources.

From the creation of this standard, emerged a lot of research from many countries around the world that created instances of all kind of types, some representing real cases from several countries, for example, England [26], Finland [16], Greek [25], Netherlands [6] and Brazil [22,23]. All these instances were published in a global and free public repository, to be used as benchmarking for other studies, such as the one conducted in this paper.

2.2 KHE Library

In 2006, [11] presented a library, called KHE (*Kingston High School Timetabling Engine*). This library was created exclusively for school timetabling problems, with the objective of facilitating and optimizing the management of the instances and their solutions. Completely integrated with the XHSTT standard, the main points of the use of this library are the data structures available and the possibility of using the function of generating initial solutions, called KheGeneralSolve. This routine generates an initial solution in a fast and easy fashion, even in large and complex instances. This library is available on the Internet and can be used freely for studies and researches in this area. Its creator also provides a service to evaluate solutions, called HsEval, available at http://www.it.usyd.edu.au/~jeff/cgi-bin/hseval.cgi.

2.3 ITC 2011

With this standards and variety of libraries for handling them, the PATAT members launched the third edition of an International Timetabling Competition -

ITC, dedicated to High School timetabling problems, in 2011. This was the last and most recent competition in this area. The two previous ones were organized in 2002 and 2007, with others specific themes of timetabling.

The ITC 2011 was composed by three phases:

- Phase 1: the instances were published and the competitors were responsible to generate the best solutions without restrictions of time and computational resources;
- Phase 2: the organizers were responsible for executing each algorithm under the same conditions, using instances not previously known and having a time limit of 1000 s of processing;
- Phase 3: the competitors generated the solutions in a set of hidden instances and, as in the phase one, it was not defined time and technology restrictions. Only the top five competitors of Phase 2 participated of this phase.

3 Proposed Approaches

In order to solve the School Timetabling Problem, in the current article we propose two algorithms for solving it, both of them based on the Variable Neighborhood Search (VNS) metaheuristic [13].

The first one, called Adaptive VNS, is described in Subsect. 3.1, and the second one, named SGVNS, is presented in Subsect. 3.2. Finally, in Subsect. 3.3, the types of moves used to explore the solution space of the problem are detailed.

3.1 Adaptive VNS

The proposed Adaptive VNS algorithm is a variant of the classic VNS metaheuristic, in which the used neighborhoods for local searches are chosen according to evolving probabilities.

This approach is similar to that presented in [1]. The basic principle is that the neighborhoods that generate better solutions should have probabilities higher than the other ones that are not generating good solutions at that moment. In order to avoid premature convergence of the algorithm, and avoiding getting biased to some neighborhoods, whenever a better solution is found, the probabilities are periodically reset.

The implementation follows the pseudo-code presented at Algorithm 1. Initially, in line 10, all $|N|$ neighborhoods used for local searches have the same probability of being chosen, that is, the parameter $probneighborhood(N^l)$ is set to $1/|N|$ (0.2 in our case). On the loop started at line 14 a shaking move using Kempe's chain neighborhood is applied during $k_{current}$ times. The neighborhood used to perform local search according to the current probabilities is chosen at line 18. As in timetabling problems there are many plateaus, solutions with an evaluation less than or equal to that of the current solution are accepted (line 20 of Algorithm 1). The probabilities of all neighborhoods are recalculated every *itercalc* iterations (line 40 of Algorithm 1).

Algorithm 1. Adaptive VNS

Input: Initial solution s_0; Maximum runtime (*MaxTime*); Maximum number of moves of the Kempe's Chain; Iterations for recalculating the probabilities (*itercalc*); set of $|N|$ neighborhoods N; Number of recalculating probabilities without improvement to restart probabilities (*IterRestart*).
Output: Best solution s.
begin
  $s \leftarrow s_0$;
  $s' \leftarrow s_0$;
  $improvement \leftarrow 0$;
  $k_{current} \leftarrow 1$;
  $number_{itercalc} \leftarrow 1$;
  for *each neighborhood* N^l *of* N **do**
    $probneighborhood(N^l) \leftarrow 1/|N|$;
  end
  iter $\leftarrow 1$;
  while $time \leq MaxTime$ **do**
    for $k = 0$; $k < k_{current}$ **do**
      $s' \leftarrow$ neighbor of s' built by applying the Kempe's Chain move;
    end
    $prob \leftarrow$ random number between 0 and 1;
    $l \leftarrow$ chosen neighborhood N^l according to the probability $probneighborhood(N^l)$ and random number $prob$;
    $s' \leftarrow$ Local Search using neighborhood $N^l(s')$;
    if $f(s') \leq f(s)$ **then**
      $s \leftarrow s'$;
      $k_{current} \leftarrow 1$; $improvement \leftarrow 1$;
    end
    else
      $s' \leftarrow s$;
      if $k_{current} \leq Kempe_{\max}$ **then**
        $k_{current} \leftarrow k_{current} + 1$;
      end
    end
    if *rest of division of iter by* itercalc *is* 0 **then**
      $number_{itercalc} \leftarrow number_{itercalc} + 1$;
      if $number_{itercalc} \geq IterRestart$ *and* $improvement = 0$ **then**
        for *each neighborhood* N^l *of* N **do**
          $probneighborhood(N^l) \leftarrow 1/|N|$;
        end
        $number_{itercalc} \leftarrow 1$;
      end
      else
        for *each neighborhood* N^l *of* N **do**
          update $probneighborhood(N^l)$
        end
      end
      $improvement \leftarrow 0$;
      iter $\leftarrow$ iter $+1$;
    end
  end
end
return s

3.2 Skewed GVNS (SGVNS)

This algorithm was built by merging two variations of VNS metaheuristics: Skewed VNS (SVNS) and General VNS (GVNS).

The SVNS is a variation of VNS proposed by [13]. It uses a parameter α to accept solutions that are worse than the current solution. The concept involved is that better solutions can be far away from the current solution, so it is necessary to go through intermediate (and worse) steps to reach them.

On the other hand, the GVNS algorithm, proposed in [14], uses Variable Neighborhood Descent (VND) algorithm to perform local searches. VND [9] is a descent method that uses systematic changes of neighborhoods to explore the space solution. It returns a local optimum among all the used neighborhoods.

The proposed SGVNS uses also VND as a local search method. In addition, as in the SVNS algorithm, a parameter α is used to accept intermediate solutions that are worse than the current solution.

Algorithms that accept worse solutions can bring a problem of execution, called cycling. It occurs when the algorithm remains stuck in the same sequence of solutions. To avoid this behavior, a Tabu List was implemented in a way that a short time list stores the values of solutions already visited. In consequence, the algorithm prevents the same sequence of solutions from being generated again. This Tabu List has a length defined by a parameter and works with FIFO protocol (First In First Out) that means that when the length is achieved the first value is overwritten by the next and so on.

The pseudo-code of SGVNS is described in Algorithm 2. At line 14 it is verified if the new solution will be considered or not according to the parameter α and the list of solution values already generated. As in the Adaptive VNS algorithm, solutions with evaluation less than or equal to the current solution are accepted (line 18 of the SGVNS Algorithm).

3.3 Moves

Both algorithms use the Kempe's Chain move for shaking the current solution. This move was proposed in [10] to the graph coloring problem. It is based on the concept that some changes in the solution can generate infeasible solutions, creating conflicts and in order to remove them it is necessary to perform a sequence of other moves. These modifications in sequence applied to a determined solution are called Kempe's Chain.

When the solution does not improve, both algorithms increase the number of times that the Kempe's Chain move is executed until a limit value defined by the parameter $Kempe_{\max}$. When an improved solution is found, each algorithm returns to its initial configuration and only one Kempe's Chain move is performed. This strategy has the objective to search better solutions and not get stuck in local optimums.

The SGVNS algorithm uses the classic VND algorithm to perform local searches and it returns the optimum in relation to all neighborhoods. The VND

Algorithm 2. SGVNS

```
1  Input: Initial solution (s_0); Maximum runtime (MaxTime); Maximum
   number of Kempe's Chain move (Kempe_max); percentage to accept worse
   solutions (α); Length of the Tabu List.
2  Output: Improved solution s found.
3  begin
4      s ← s_0;
5      s' ← s_0;
6      s_temp ← s;
7      k_current ← 1;
8      Insert f(s) in Tabu List;
9      while time ≤ MaxTime do
10         for k = 0; k < k_current do
11             s' ← neighbor of s' using Kempe's Chain move;
12         end
13         s' ← Local search using VND algorithm(s');
14         if ((f(s') ≤ ((1 + α) × f(s))) and (f(s') ∉ Tabu List)) then
15             s_temp ← s';
16             k_current ← 1;
17             Insert f(s') in Tabu List;
18             if (f(s') ≤ f(s)) then
19                 s ← s';
20             end
21         end
22         else
23             s' ← s_temp;
24             if k_current ≤ Kempe_max then
25                 k_current ← k_current + 1;
26             end
27         end
28     end
29 end
30 return s
```

algorithm is described in the literature, so it is not presented in this article. The neighborhoods are generated with one of the moves described below:

Event Swap: this move consists in selecting two lessons and changing the time slots between them;

Event Move: this move consists in choosing one lesson and moving it to another time slot that is empty;

Event Block Swap: like to the Event Swap, it consists in swapping the time slot of two lessons. However, if the lessons have different durations, one lesson is moved to the last time slot occupied by the other lesson. That is, if one of the selected lessons has another lesson in a time slot adjacent to it, the change involves both lessons, not only the selected one. This move allows contiguous time slots to be exchanged;

Move Time: in this move, two classes are chosen and exchanged;
Change Time: this move consists in choosing one class and changing its resource with another resource that is current available.

On the other hand, the Adaptive VNS algorithm chooses, in a probabilistic fashion, only one of these moves described above to perform only **one** local search.

4 Computational Experiments

Both algorithms were implemented in C++ using the IDE Code::Blocks. All tests were done in a notebook with Intel Core i5 processor, 4 GB RAM memory running Windows 10.

In order to test the algorithms, instances from ITC 2011 were used. Among the twenty-one instances presented in that event, five of them already were in local optimum since the initial solution, so it was not necessary to work with them. In the first phase of ITC 2011, although there were no processing time restrictions, the computational time limit for each instance used by Goal Solver algorithm [5] (the winner of the competition) was 1000 s. In the current experiments, the same value of computational time limit to each instance was considered. The initial solutions were provided by the organizers and they were made available together with the instances, in the same XHSTT file. The three instances from Australia (AustraliaBGHS98, AustraliaSAHS96 and AustraliaTES99) presented worse initial solutions compared with that generated by KHE library. Thus, in these instances, we used the solutions generated by KHE.

4.1 Parameter Tuning

Several distinct experiments were conducted to find the best set of parameter configurations for each algorithm. The iRace package (http://iridia.ulb.ac.be/irace/) was used for the accomplishment of this task. This tool implements the iterated racing procedure [12] and it is an extension of the iterated F-race (I/F-Race) proposed by [4]. The main function of iRace is the automatic configuration of optimization algorithms in order to determine the most appropriate parameter settings for an optimization method. The iRace framework is implemented as an R package [18] and builds upon the race package.

The iRace analysis was done on a budget of 3,000 runs for each algorithm (Adaptive VNS and SGVNS). Due to the high duration of the tests, we do not use 1,000 s as a stopping criterion in this phase. Three different times were used as stopping criterion for each algorithm applied in each instance: 10 s, 30 s and 60 s. Tables 1 and 2 show the parameters tested by iRace for both the SGVNS and the Adaptive VNS algorithms, respectively.

As a result, the iRace indicated the best parameters for each method, as shown in the values depicted at Table 3. In addition to the parameter calibration, the iRace gave us the feedback that the performance of both algorithms improved

Table 1. The parameters tested by iRace in SGVNS algorithm.

Parameter	Values				
α	0.001	0.01	0.025	0.05	0.075
$Kempe_{\max}$	1	5	10	-	-
Length of the Tabu List	5	7	10	15	-

Table 2. The parameters tested by iRace in Adaptive VNS algorithm.

Parameter	Values			
Iterations to restart probabilities	5	10	15	20
$Kempe_{\max}$	1	5	10	-
itercalc	100	250	500	750

from 10 s to 60 s. In this sense, the best parameters were picked from executions with 60 s, which are the ones that provide more liberty to the methods to play with exploration-exploitation concepts.

4.2 Results

The same set of neighborhoods N = {Event Swap, Event Move, Event Block Swap, Move Time and Change Time} was used for both algorithms.

Table 4 shows the best results obtained by the algorithms Goal Solver of [5], GVNS of [24] and the proposed SGVNS and Adaptive VNS algorithms. In addition, Table 5 shows the average results also obtained by these same algorithms. In both tables, the first column shows the tested instances and the second one, the value of the initial solutions provided by the organizers of ITC 2011, except for instances AustraliaBGHS98, AustraliaSAHS96 and AustraliaTES99, whose initial solutions were generated by the KHE algorithm of [11]. The following columns present the values of Goal Solver, GVNS, SGVNS e Adaptive VNS algorithms, respectively.

Each instance was executed 30 times for each algorithm. It is noteworthy that all algorithms were executed on the same machine, and the Goal Solver code was provided by its developers.

The values presented in each cell of Tables 4 and 5 are pairs x/y, where x means the sum of penalties for hard constraints not met and y the sum of penalties for soft constraints not met. In case of a tie in the penalties for hard constraints not met, the solutions that have smallest soft constraints not met are considered the best ones. A value highlighted in **bold** means that is considered to be the best result produced among all the algorithms.

Analyzing the best results as presented in Table 4, it is verified that SGVNS outperforms the other algorithms in most instances. On the other hand, the Adaptive VNS algorithm did not outperform the other algorithms in any instance, although it has produced good results as well. Considering the sixteen

Table 3. Best parameters indicated by iRace.

SGVNS	α	$Kempe_{\max}$	Length of the Tabu List	Execution time
	0.025	5	10	60
Adaptive VNS	*itercalc*	$Kempe_{\max}$	Iterations to restart probabilities	Execution time
	500	1	5	60

Table 4. Best results of the algorithms.

Instance	KHE (Initial Solution)	Goal Solver (SA + ILS)	GVNS	SGVNS	Adaptive VNS
AustraliaBGHS98	6/450	6/450	4/370	**1/401**	7/431
AustraliaSAHS96	17/55	14/50	**12/51**	13/46	17/52
AustraliaTES99	7/163	7/161	**7/151**	7/163	7/163
BrazilInstance1	0/24	0/12	**0/11**	**0/11**	0/12
BrazilInstance4	0/112	0/91	0/94	**0/90**	0/94
BrazilInstance5	0/225	0/164	0/158	**0/149**	0/165
BrazilInstance6	0/209	0/149	0/148	**0/131**	0/163
BrazilInstance7	0/330	0/264	0/249	**0/248**	0/282
EnglandStPaul	0/18,444	0/18,092	0/12,542	**0/12,466**	0/18,418
FinlandHighSchool	**0/1**	**0/1**	**0/1**	**0/1**	**0/1**
FinlandSecondarySchool	0/106	**0/86**	0/87	0/88	0/87
ItalyInstance1	0/28	0/19	**0/18**	**0/18**	**0/18**
NetherlandsGEPRO	1/566	1/566	**1/434**	1/441	1/532
NetherlandsKottenpark2003	0/1,410	0/1,409	**0/1,216**	0/1,281	0/1,372
NetherlandsKottenpark2005	0/1,078	0/1,078	0/881	**0/877**	0/1,078
SouthAfricaLewitt2009	0/58	**0/22**	0/24	0/24	0/42

instances, SGVNS algorithm reached the best results in ten ones and GVNS in seven ones. Goal algorithm, in turn, reached the best results only in three instances.

In another analysis, focusing in the average of the results, as presented in Table 5, SGVNS algorithm reached the best results in six instances and GVNS in eight ones. Goal algorithm reached the best results in five instances and the adaptive VNS in only one instance.

In order to evaluate if there were significant differences among the algorithms, the R Studio tool was used to perform the statistical analyzes of the results. For this analysis all samples were used and it was concluded that the samples did not present normal distribution applying the Shapiro-Wilk test [15]. Then, a non-

Table 5. Average results of the algorithms.

Instance	KHE (Initial Solution)	Goal Solver (SA + ILS)	GVNS	SGVNS	Adaptive VNS
AustraliaBGHS98	6/450	6/450	5/450	**3/514**	7/431
AustraliaSAHS96	17/55	**16/20**	16/30	16/91	17/53
AustraliaTES99	7/163	**7/162**	**7/162**	7/163	7/163
BrazilInstance1	0/24	0/14	**0/11**	**0/11**	0/13
BrazilInstance4	0/112	**0/98**	0/100	0/100	0/099
BrazilInstance5	0/225	0/181	0/178	**0/177**	0/188
BrazilInstance6	0/209	0/168	**0/160**	0/170	0/175
BrazilInstance7	0/330	0/280	**0/276**	0/289	0/300
EnglandStPaul	0/18,444	0/18,444	**0/14,217**	0/14,442	0/18,418
FinlandHighSchool	**0/1**	**0/1**	**0/1**	**0/1**	**0/1**
FinlandSecondarySchool	0/106	**0/89**	0/92	0/93	0/90
ItalyInstance1	0/28	0/21	0/21	**0/19**	0/24
NetherlandsGEPRO	1/566	1/566	**1/446**	1/484	1/551
NetherlandsKottenpark2003	0/1,410	0/1,409	**0/1,290**	0/1,387	0/1,377
NetherlandsKottenpark2005	0/1,078	0/1,078	**0/956**	0/1,056	1/1,078
SouthAfricaLewitt2009	0/58	0/30	0/30	**0/28**	0/48

parametric test was used, the Friedman test [15], with the goal of verifying if the algorithms had significant differences among them. The test returned a p-value of 0.0003287. Thus, considering a significance level of 95%, the algorithms had statistically significant differences among them.

From this result, we proceed the pairwise Wilcoxon test [15] to evaluate if there is statistical difference between each pair of algorithms. The results indicated that the SGVNS is statistically different from all others algorithms. The GVNS is also statistically different from Adaptive VNS. The other comparisons are statistically equivalent. It was used the BH p-value adjustment method [3].

5 Conclusions

This paper proposed two metaheuristic approaches for solving the School Timetabling problem, both based on the VNS metaheuristic. The first algorithm represents a combination between GVNS and SVNS metaheuristics and it was called SGVNS or Skewed GVNS. The second one, named Adaptive VNS, is an adaptive approach based on VNS that defines the neighborhood to perform local searches by means of probabilities, and prioritizes the neighborhoods that have obtained the best results in past iterations.

Both algorithms produced good solutions to this problem with an advantage of the SGVNS algorithm that is statistically different from all other algorithms. In turn, the Adaptive VNS is equivalent to the Goal Solver of [5]. Considering the best results, SGVNS performed equal or better than the Goal Solver and the GVNS algorithm of [24] in ten from sixteen instances.

As future work, we suggest:

- optimize the calculation of the probabilities of the Adaptive VNS algorithm;
- Evaluate both algorithms in other instances, such as those used in the second phase of ITC 2011.

Acknowledgments. We would like to thank CAPES, CNPq, FAPEMIG, CEFET-MG and UFOP for supporting the development of this work. We also thank the authors of the Goal Solver algorithm for making its source code available.

References

1. Aziz, R.A., Ayob, M., Othman, Z., Ahmad, Z., Sabar, N.R.: An adaptive guided variable neighborhood search based on honey-bee mating optimization algorithm for the course timetabling problem. Soft Comput. **21**(22), 6755–6765 (2017)
2. Bardadym, V.A.: Computer-aided school and university timetabling: the new wave. In: Burke, E., Ross, P. (eds.) PATAT 1995. LNCS, vol. 1153, pp. 22–45. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61794-9_50
3. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Stat. Soc. Ser. B (Methodol.) **57**, 289–300 (1995)
4. Birattari, M., Balaprakash, P., Dorigo, M.: The ACO/F-Race algorithm for combinatorial optimization under uncertainty. In: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R.F., Reimann, M. (eds.) Metaheuristics. ORSIS, vol. 39, pp. 189–203. Springer, Boston, MA (2007). https://doi.org/10.1007/978-0-387-71921-4_10
5. da Fonseca, G.H.G., Santos, H.G., Toffolo, T.Â.M., Brito, S.S., Souza, M.J.F.: GOAL solver: a hybrid local search based solver for high school timetabling. Ann. Oper. Res. **239**(1), 77–97 (2016)
6. de Haan, P., Landman, R., Post, G., Ruizenaar, H.: A case study for timetabling in a dutch secondary school. In: Burke, E.K., Rudová, H. (eds.) PATAT 2006. LNCS, vol. 3867, pp. 267–279. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77345-0_17
7. Even, S., Itai, A., Shamir, A.: On the complexity of time table and multi-commodity flow problems. In: Proceedings of the 16th Annual Symposium on Foundations of Computer Science, pp. 184–193 (1975)
8. Gotlieb, C.C.: The construction of class-teacher timetables. In: Proceedings of IFIP Congress, pp. 73–77 (1963)
9. Hansen, P., Mladenovic, N., Pérez, J.A.M.: Variable neighborhood search: methods and applications. 4OR: Q. J. Belg. Fr. Ital. Oper. Res. Soc. **6**, 319–360 (2008)
10. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. Oper. Res. **39**(3), 378–406 (1991)

11. Kingston, J.H.: Hierarchical timetable construction. In: Burke, E.K., Rudová, H. (eds.) PATAT 2006. LNCS, vol. 3867, pp. 294–307. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77345-0_19
12. Lopez-Ibanez, M., Dubois-Lacoste, J., Stutzle, T., Birattari, M.: The irace package: iterated racing for automatic algorithm configuration. IRIDIA, Universite Libre de Bruxelles, Belgium, Technical report, TR/IRIDIA/2011-004 (2011)
13. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
14. Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., Čangalović, M.: General variable neighborhood search for the continuous optimization. Eur. J. Oper. Res. **191**, 753–770 (2008)
15. Montgomery, D.C.: Design and Analysis of Experiments. Wiley, Boco Raton (2017)
16. Nurmi, K., Kyngas, J.: A framework for school timetabling problem. In: Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications, Paris, pp. 386–393 (2007)
17. Post, G., et al.: XHSTT: an XML archive for high school timetabling problems in different countries. Ann. Oper. Res. **218**, 295–301 (2014)
18. R Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2014). http://www.R-project.org/
19. Schaerf, A.: A survey of automated timetabling. Artif. Intell. Rev. **13**(2), 87–127 (1999)
20. Santos, H.G., Ochi, L.S., Souza, M.J.F.: A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. J. Exp. Algorithmics (JEA) **10**, 2–9 (2005)
21. Santos, H.G., Souza, M.J.F.: Timetabling in educational institutions: formulations and algorithms (in Portuguese). In: Proceedings of the XXXIX Brazilian Symposium of Operations Research, Fortaleza, Brazil, pp. 2827–2882 (2007)
22. Souza, M.J.F.: School timetabling: an approximation by metaheuristics (in Portuguese). Ph.D. thesis, Programa de Pós-graduação em Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Brazil (2000)
23. Souza, M.J.F., Maculan, N., Ochi, L.S.: A GRASP-Tabu search algorithm for solving school timetabling problems. In: METAHEURISTICS: Computer Decision-Making. Kluwer Academic Publishers, Dordrech, vol. 86, pp. 659–672 (2004)
24. Teixeira, U.R., Souza, M.J.F., de Souza, S.R.: A local search approach using GVNS for solving school timetabling problems (in Portuguese). In: Proceedings of the XXXVIII Ibero Latin American Congress on Computational Methods in Engineering (CILAMCE), Florianópolis, Brazil (2017). https://doi.org/10.20906/CPS/CILAMCE2017-1240
25. Valouxis, C., Housos, E.: Constraint programming approach for school timetabling. Comput. Oper. Res. **30**(10), 1555–1572 (2003)
26. Wright, M.: School timetabling using heuristic search. J. Oper. Res. Soc. **47**(3), 347–357 (1996)